VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

ZAVEDENÍ AGILNÍCH METOD VE FIRMĚ

APPLICATION OF AGILE IN A COMPANY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PETR FOTIJEV
AUTHOR

VEDOUCÍ PRÁCE — Ing. LENKA SMOLÍKOVÁ, Ph.D.
SUPERVISOR

BRNO 2016

Tato verze diplomové práce je zkrácená (dle Směrnice děkana č. 2/2013). Neobsahuje identifikaci subjektu, u kterého byla diplomová práce zpracována (dále jen „dotčený subjekt“) a dále informace, které jsou dle rozhodnutí dotčeného subjektu jeho obchodním tajemstvím či utajovanými informacemi.

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2015/2016
Ústav informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Fotijev Petr, Bc.

Informační management (6209T015)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává diplomovou práci s názvem:

Zavedení agilních metod ve firmě

v anglickém jazyce:

Application of Agile in a Company

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Návrh řešení a přínos návrhů řešení
Závěr
Seznam použité literatury

Seznam odborné literatury:

BUCHALCEVOVÁ, A. Metodiky budování informačních systémů. 1.vyd.Praha: Oeconomica, 2009. 205 s. ISBN 978-80-245-1540-3.
DOLEŽAL, J., P. MÁCHAL a B. LACKO. Projektový management podle IPMA. 1. vyd. Praha: Grada, 2009. 507 s. ISBN 978-80-247-2848-3.
PROCHÁZKA J. a C. KLIMEŠ. Provozujte IT jinak: Agilní a štíhlý provoz, podpora a údržba informačních systémů a služeb. Praha: Grada Publishing, 2011. 288 s. ISBN 978-80-247-4137-6.
RUBIN, K. S., Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley Professional,2012. 452 s. ISBN 978-01-370-4329-3.
ŠOCHOVÁ, Z. a E. KUNCE. Agilní metody řízení projektů. 1. vyd. Brno: Computer Press, 2014. 175 s. ISBN 978-80-251-4194-6.

Vedoucí diplomové práce: Ing. Lenka Smolíková, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2015/2016.

L.S.

doc. RNDr. Bedřich Půža, CSc.	doc. Ing. et Ing. Stanislav Škapa, Ph.D.
Ředitel ústavu	Děkan fakulty

V Brně, dne 30.11.2015

Abstrakt

Diplomová práce se zabývá problémy projektů informačních technologií a jejich řešením pomocí agilních metod řízení projektů. Na základě analýzy společnosti a jejich projektů je navrženo řešení vývoje pomocí agilních metod.

Abstract

This master thesis deals with problems of information technology projects and their solutions using agile project management. Based on an analysis of the company and its projects is suggested solution using agile methods.

Klíčová slova

Řízení projektů, Agilní metody, Scrum, Kanban, Vývoj software

Key words

Project management, Agile methods, Scrum, Kanban, Software development

Bibliografická citace

FOTIJEV, P. *Zavedení agilních metod ve firmě*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2016. 69 s. Vedoucí diplomové práce Ing. Lenka Smolíková, Ph.D..

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů jsou úplné a že jsem ve své práci neporušil autorské práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

Bc. Petr Fotijev

V Brně dne 20. ledna 2016 .....

Podpis

Poděkování

Chtěl bych na tomto místě poděkovat především vedoucí mé diplomové práce Ing. Lence Smolíkové, Ph.D. za ochotu, trpělivost a cenné rady při psaní této práce. Také bych chtěl poděkovat mé rodině a přátelům za podporu a pochopení během studia.

Obsah

Úvod

Projektový management ve své podstatě existoval již v době pyramid, byl zdokonalen během druhé světové války, a přestože jsme díky němu poslaly raketu na měsíc, tyto postupy již nestačí potřebám moderního vývoje informačních technologií. Společnost, která se chce udržet na trhu s informačními technologiemi, musí na nové trendy a měnící se požadavky reagovat. Využití agilních metod by to mělo umožnit.

Práce je rozdělena do tří částí, přičemž v první jsou zpracována teoretická východiska práce. Zabývá se důvody neúspěšnosti projektů informačních technologií a používanými metodami řízení projektů, jak klasických, tak i nových agilních.

V druhé kapitole se analyzuje společnost, jak její produkty, které jsou právě vyvíjeny, tak i ty které se nachází již v poprojektové fázi a společnost je udržuje. Dále se analyzují postupy vývoje softwaru používané společností.

Poslední kapitola obsahuje návrh řešení pro zavedení agilních metod do společnosti, jejich užití v novém projektu a také udržování již dokončených informačních systémů.

Cíle práce, metody a postupy zpracování

Cíl práce

Cílem diplomové práce je prostudovat problematiku agilních metod v projektovém managementu, zvolit vhodnou metodu a navrhnout její využití pro řízení projektu vývoje nové generace informačního systému a ostatních projektů informačních systémů v poprojektové fázi.

Metody zpracování

Práce je zpracována využitím kombinací agilních metod Scrum a Kanban.

Postupy zpracování

V kapitole Teoretická východiska práce jsou popsány nejpoužívanější metody pro softwarový vývoj. Následně je zpracována analýza společnosti na základě informací získaných během pracovního poměru. Navržené řešení bylo vytvořeno na základě poznatků z obou předchozích částí.

1 Teoretická východiska práce

1.1 Řízení projektů

Potřeba projektového managementu vzniká po druhé světové válce omezením zdrojů a času pro tvorbu projektu. Dalším faktorem je zrychlování doby v souvislosti s rozvojem komunikačních kanálů a rozvojem informačních technologií. Projektové řízení se stalo nástrojem realizace potřebných změn, které podniky začaly postupně realizovat. Přitom nešlo jen o změny organizační nebo technologické, ale vývoj nových produktů a dosažení velmi ambiciózních cílů jako přistání člověka na měsíci. (3 str. 35)

Rozšiřování využití informačních technologií se stává hnací sílou rozvoje projektového řízení, které se do té doby využívalo primárně v průmyslu a stavebnictví. V oblasti IT je využíváno v hardwarových projektech, které jsou v podstatě velmi podobné průmyslovým projektům, ale také ve vývoji složitých aplikací, kde řídí složitý tvůrčí proces, nebo koordinaci obojího. V současné době může díky projektovému managementu spolupracovat na projektu tým lidí žijících kdekoliv na světě. (3 str. 35)

1.2 Specifika softwarových projektů

Využívání počítačů roste bezprecedentní rychlostí, a to napříč všemi spektry každodenního života. Jsou využívány například v pokladnách, bankomatech, ledničkách, pro online obchodování, objednávání lístků, účetnictví, medicíně a dalších. Vývoj softwaru je velmi komplikovaný proces zahrnující mnoho stakeholderů, kteří mají své vlastní zájmy, proto úspěch softwarového projektu zahrnuje množství různých aspektů. Na rozdíl od projektů v ostatních oborech, při vývoji software lze cíle dosáhnout velkým množstvím způsobů, navíc žádné dva projekty nejsou stejné. (6 str. 541)

Často se měnící zadání, různě velké investice a napojení na další obory zvyšuje riziko neúspěchu těchto projektů. Kromě standardních metrik projektů se objevují

další: množství dat, rychlost odezvy, přesnost očekávaného výsledku, které dále zvyšují komplexnost softwarového vývoje, navíc se očekává, že výsledný systém bude spolehlivý a flexibilní. (6 str. 541)

Mezi hlavní problémy softwarových projektů patří:

- **Složité odhadování časové náročnosti** – časový odhad není možné udělat s 0 % odchylkou, nejpřesnější týmy mají odchylku v řádu jednotek procent za předpokladu krátkých iteracích, malých úkolů, časté spolupráce se zákazníkem, zkušeného stabilního týmu a práce na velmi podobných projektech. Ve chvíli, kdy tyto faktory nejsou splněny, může odchylka narůst v desítkách procent.
- **Nejasné nebo nekompletní zadání** – to, co zákazník potřebuje, leží v jeho hlavě, záleží také na schopnostech týmu mu navrhnout kreativní řešení, které sám neviděl a budoucnosti, určující jak se změní podmínky na trhu nebo technologie, než se projekt stihnete dokončit.
- **Lidský faktor** – i přes použití frameworků, nastavení štábní kultury a pečlivého testování, není možné odstranit z procesu lidský faktor, ač se jedná o prodlužování z důvodů chyb nebo záměny vývojáře v týmu. (6 str. 542)

1.3 Neúspěch softwarových projektů

Ve spojených státech se každoročně investuje do vývoje počítačových aplikací 250 miliard dolarů rozdělených mezi přibližně 175 000 projektů. Průměrná cena projektu ve velké společnosti je 2 322 000 dolarů, ve střední 1 331 000 dolarů a v malé 434 000 dolarů. Většina těchto projektů je odsouzena k neúspěchu. Průzkum skupiny The Standish Group ukázal, že závratných 18 % projektů bude zrušeno před dokončením. Další výsledky ukazují, že 43 % projektů se prodraží oproti jejich původním rozpočtům nebo bude jejich vývoj prodloužen oproti původním časovým odhadům. (14 str. 3)

Podle studie může projekt skončit úspěšně, když je dokončen v daném časovém rozsahu, realizace nepřesáhne stanovený rozpočet a je funkční podle zadané specifikace. Přestože je projekt dokončen, je úspěšnost projektu napadnutelná, pokud jedna nebo více z předchozích podmínek není splněna. Projekt může být také zastaven před dokončením z důvodu změny zadání, technologie nebo nespokojenosti s jeho průběhem. (13 str. 5)

Tabulka 1: Úspěšnost projektů IT
(Zdroj: 13 str. 5)

	2004	2006	2008	2010	2012
Úspěšné	29 %	35 %	32 %	37 %	39 %
Napadené	53 %	46 %	44 %	42 %	43 %
Neúspěšné	18 %	19 %	24 %	21 %	18 %

Z výzkumu vyplývá, že pouze 20 % funkcí programů je používáno často, 30 % je využíváno občas nebo nepravidelně a zbylých 50 % je používáno zřídka nebo vůbec. Přičemž je známo, že získávání požadavků, výběr a jejich implementace je nejsložitější věc při vývoji zakázkového softwaru. Z toho bez pochyb plyne, že zaměřením se na 20 % často používaných funkcí se získá 80 % hodnoty aplikace a dojde k maximalizaci investice a zvýšení celkové spokojenosti uživatele. (13 str. 5)

1.3.1 Statistiky neúspěchu

Statistika nárůstu rozpočtů neúspěšných a rozporovaných projektů ukazuje, že 52,7 % projektů se prodraží o 189 % oproti jejich původním rozpočtům, bez zahrnutí nákladů příležitosti. (14 str. 5)

Tabulka 2: Navýšení rozpočtu projektů IT
(Zroj: 14 str. 6)

Zvýšení ceny	Procento respondentů
pod 20 %	15,5 %
21-50 %	31,5 %
51-100 %	29,6 %
101-200 %	10,2 %
201-400 %	8,8 %
přes 400 %	4,4 %

Průměrně je doba realizace neúspěšných a rozporovaných projektů prodloužena o 222 % oproti původnímu předpokladu. (14 str. 6)

Tabulka 3: Prodloužení projektů IT
(Zdroj: 14 str. 6)

Prodloužení času	Procento respondentů
pod 20 %	13,9 %
21-50 %	18,3 %
51-100 %	20,0 %
101-200 %	35,5 %
201-400 %	11,2 %
přes 400 %	1,1 %

Průměrně pouze 61% funkcí původně specifikováno při zadání projektu bylo přítomno v konečném řešení. (14 str. 7)

Tabulka 4: Splněná funkčnost projektů IT
(Zdroj: 14 str. 7)

Funkčnost	Procento respondentů
pod 25 %	4,6 %
25-49 %	27,2 %
50-74 %	21,8 %
75-99 %	39,1 %
100 %	7,3 %

1.3.2 Profily úspěšnosti a neúspěšnosti projektů

Nejdůležitější je zjistit důvod, proč jsou projekty informačních technologií neúspěšné. V rámci průzkumu byli dotázáni výkonní IT manažeři, na jejich názor co je nutné pro úspěšný projekt. Na prvních místech skončilo zapojení uživatelů, podpora managementu a jasně definované požadavky, nejasnosti v těchto kritérií velmi zvyšují pravděpodobnost neúspěchu projektu. (14 str. 8)

Tabulka 5: Faktory úspěchu projektů IT
(Zdroj: 14 str. 8)

Faktory napadení projektů	Procento respondentů
Zapojení uživatelů	15.9 %
Podpora týmu managementem	13.9 %
Jasně definované požadavky	13.0 %
Dobré plánování	9.6 %
Realistické očekávání	8.2 %
Malé milníky	7.7 %

Kompetentní tým	7.2 %
Vedení	5.3 %
Jasná vize a cíle	2.9 %
Tvrdá práce	2.4 %
Ostatní	13.9 %

Účastníci byli také dotázáni, co je podle nich častým důvodem napadení provedení projektů.

Tabulka 6: Faktory napadení kvality projektů
(Zdroj: 14 str. 9)

Faktory napadení projektů	Procento respondentů
Nedostatečné zapojení uživatelů	12,8 %
Nekompletní požadavky a specifikace	12,3 %
Měnící se požadavky a specifikace	11,8 %
Nedostatečná podpora managementem	7,5 %
Nezvládnutí technologií	7,0 %
Nedostatek zdrojů	6,4 %
Nerealistická očekávání	5,9 %
Nejasné cíle	5.3 %
Nerealistické časové odhady	4,3 %
Nová technologie	3,7 %
Ostatní	23.0 %

Nejčastěji udávaným důvodem vedoucím k zastavení a následnému zrušení jsou nekompletní požadavky, nedostatečné zapojení uživatelů a málo zdrojů.

Tabulka 7: Faktory zrušení projektů
(Zdroj: 14 str. 9)

Faktory zrušení projektů	Procento respondentů
Nekompletní požadavky	13,1 %
Nedostatečné zapojení uživatelů	12,4 %
Nedostatek zdrojů	10,6 %
Nerealistická očekávání	9,9 %
Nedostatečná podpora managementem	9,3 %
Měnící se požadavky a specifikace	8.7 %
Špatné plánování	8,1 %
Nechuť na problému pracovat	7,5 %
Špatný IT management	6,2 %
Technologická negramotnost	4,3 %
Ostatní	9.9 %

1.4 Klasické metody

Modely vývoje se přizpůsobovaly podmínkám na trhu s informačními systémy a používaným informačním technologiím. Starší modely spíše popisují ideový přístup k projektu jako celku. Díky své abstrakci se používají dodnes, ale pro taktické a strategické řízení projektu IS/ICT. Moderní modely jsou zpracovány do detailů, pro operativní řízení se používají s podporou informačních technologií (4 str. 68).

1.4.1 Vodopádový model

Vodopádový model patří mezi nejstarší modely. Vychází z doby, kdy poptávka na trhu projekčních prací převyšovala nabídku a každý projekt informačního systému byl brán jako jedinečné díleo na zakázku (4 str. 69).

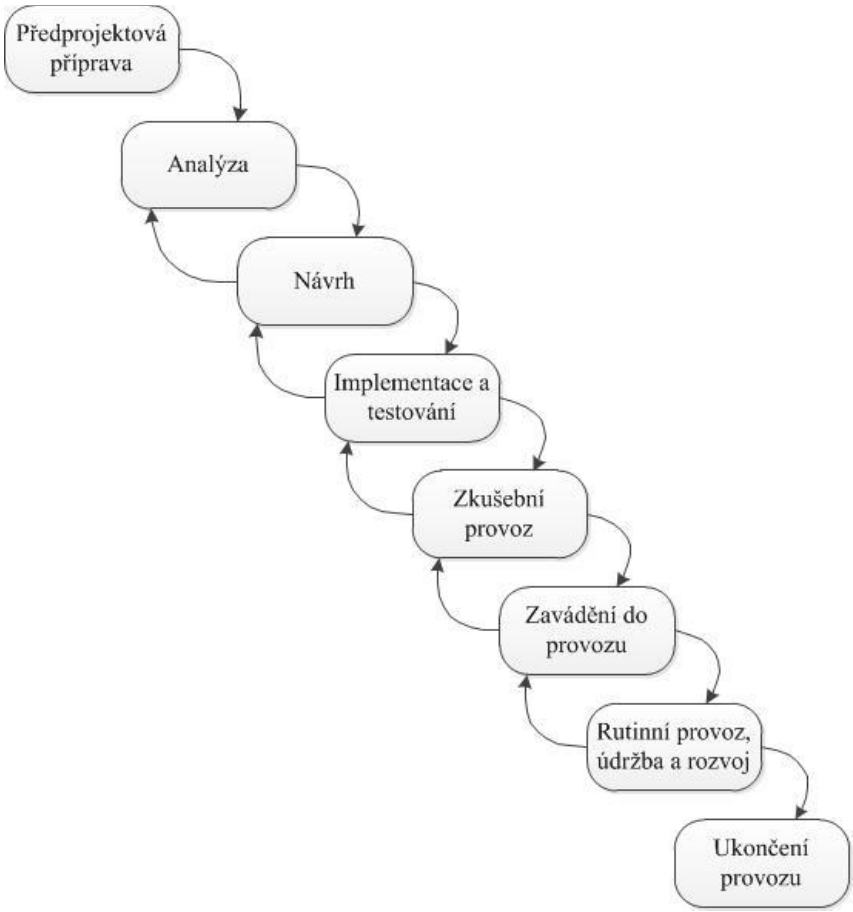
Předpoklady pro nasazení vodopádového modelu životního cyklu projektu jsou následující:

- jednotlivé etapy jsou vykonávány postupně po ukončení etapy předchozí. Výsledky etapy se po ukončení vyhodnotí a poslouží jako podklady pro etapu následující,
- nejlepší výsledek se dosáhne při dodržení pořadí na obrázku 1 (4 str. 52).

Pro ekonomické vztahy mezi dodavatelem projektu a jeho objednatelem platí:

- objednatel projektu hradí veškeré finanční náklady na tvorbu projektu informačního systému,
- riziko úspěšnosti projektu leží na objednateli,
- náklady dodavatele zahrnují vyškolení vlastních zaměstnanců a pořízení informačních technologií spojených s vývojem (4 str. 70).

Díky svému charakteru, jednoduchostí, srozumitelností, jednoduchou transformací do etap a fází do časové osy apod., se tento model používá převážně pro řízení projektu IS/ICT a sledování jeho průběhu. Model je však stále možné vidět na projektech některých manažerských aplikací. Díky ekonomickým vztahům mezi dodavatelem a objednatelem je vhodný pro firmy snažící se proniknout na trh s novým řešením (4 str. 71).

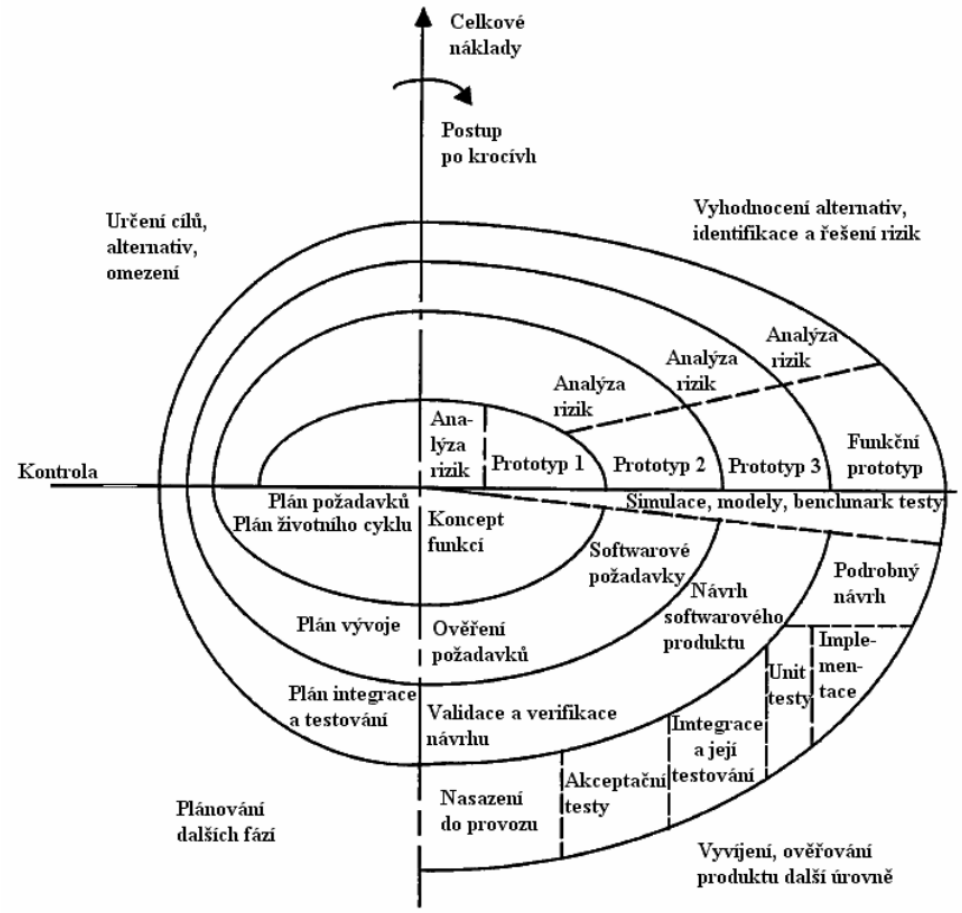


Obrázek 1: Vodopádový model životního cyklu projektu

(Zdroj: 4, str. 70)

1.4.2 Spirálový model

Tento systém vznikl v době, kdy na trhu došlo k vyrovnání mezi poptávkou a nabídkou projektů IS/ICT. Proto dodavatel začíná vyhledávat části trhu, kde existuje možnost uplatnění podpory IS/ICT. Snaží se zjistit velikost poptávky po informačním systému nebo jeho části. V případě, že zjistí možnost uplatnění v tomto odvětví trhu, vytvoří první prototyp. Ten nabídne na trhu, a pokud je prototyp kvalitní, získá prvního zájemce (4 str. 71-73; 2 str. 108).



Obrázek 2: Spirálový model

(Převzato z: http://testovanisoftwaru.cz/wp-content/uploads/2011/07/spirala.png)

Takto vznikne další prototyp obsahující požadavky prvního objednatele, ze kterých budou některé funkčnosti, které jsou nedostatkem prvního prototypu, vybrány a použity pro ostatní. Za tyto funkčnosti zaplatí objednatel méně než za funkčnosti specifické pro něj. Tento model tak vyrovnává náklady mezi objednatelem a dodavatelem (4 str. 7273; 2 str. 108).

1.4.3 Iterativní model

Iterativní model vychází ze spirálového modelu, na který implementuje vodopádový model. Hlavní myšlenkou tohoto modelu je, že člověk dokáže řešit

více malých problémů než jeden velký. Projekt se tedy rozdělí do několika menších logických částí, které na sebe navazují, tím vzniká řetěz menších vodopádů. Jednotlivé iterace jsou testovány a jejich výstupem by měla být funkční část systému. Tento přístup snižuje riziko nespokojenosti zákazníka, jelikož může po každé iteraci provést kontrolu a podat zpětnou vazbu. Na rozdíl od projektu řešeného s použitím vodopádového modelu dokáže flexibilně reagovat na změny návrhu a přidávání nových funkčností (2 str. 109).

1.5 Agilní metody

1.5.1 Manifest agilního vývoje software

Metody agilního řízení projektů se využívaly již dříve, ale pojem byl definován až na počátku roku 2001 v americkém Utahu, kde se setkali odborníci v oblastech řízení projektů a softwarového inženýrství a diskutovali o odlehčených metodách vývoje software, sepsali Manifest agilního programování, ve kterém definují přístup k vývoji známý jako agilní programování. (10 str. 14)

Jednotlivci a interakce před procesy a nástroji

Aby tým dobře fungoval, je důležité zaměřit se na jednotlivce a komunikaci mezi nimi, spíš než na definování procesů, které tým svazují a zpomalují. (10 str. 14)

Fungující software před vyčerpávající dokumentací

Softwarové firmy chtějí zákazníka většinou zaujmout rozsáhlými specifikacemi a dokumentací, které často neodpovídají skutečnosti nebo aktuálnímu stavu, způsobující zmatení zákazníka. Dokumentace je důležitá, ale měla by sloužit pro zmapování složitějších a neintuitivních oblastí, kterých by však mělo být v dobrém softwaru co nejméně. (10 str. 15)

Interní dokumentace by také neměla být zbytečně rozsáhlá, protože s její velikostí roste náročnost jejího udržování v aktuální podobě stejně jako obtížnost vyhledávání. Dokumentace vzniká jako nositel informací pro nové týmy, měla by

být stručná a obsahovat klíčové informace o architektuře produktu. Další informace je efektivnější udržovat v komentářích. (10 str. 15)

Obrázek 3: Manifest agilního vývoje software

(Zdroj: 15)

Spolupráce se zákazníkem před uzavření smlouvy

Při vývoji software je běžnou praxí, že zákazník přesně neví, co chce. Proto je nutné počítat s tím, že jeho požadavky se mohou v průběhu vývoje produktu měnit, a třeba toto zohlednit nejen ve smlouvě, ale i při spolupráci. Je nutné si uvědomit, že produkt je určen pro zákazníka a ne pro smlouvu, a tedy nevytvářet slepě podle smlouvy. Snahou vyjít vstříc novým požadavkům a pravidelnou komunikací, se může podařit zákazníka „vychovat" a vytvořit spolupracující tým, který zajistí oboustranný úspěch projektu. Takto ukončený projekt, i když ne přesně podle smlouvy, může být důvodem opětovné spolupráce, stejně tak i doporučení dalším firmám. (10 str. 16)

Pokud k takové spolupráci nedojde, může ukončení projektu provázet měsíce hádek o chybách a požadavcích, které skončí kompromisem, kdy jich půl dodělá dodavatel na vlastní náklady jako chybu a zbytek zaplatí zákazník jako novou funkčnost. V nejhorším případě začnou oba účastníci komunikovat přes právní zástupce a sejdou se u soudu. (10 str. 17)

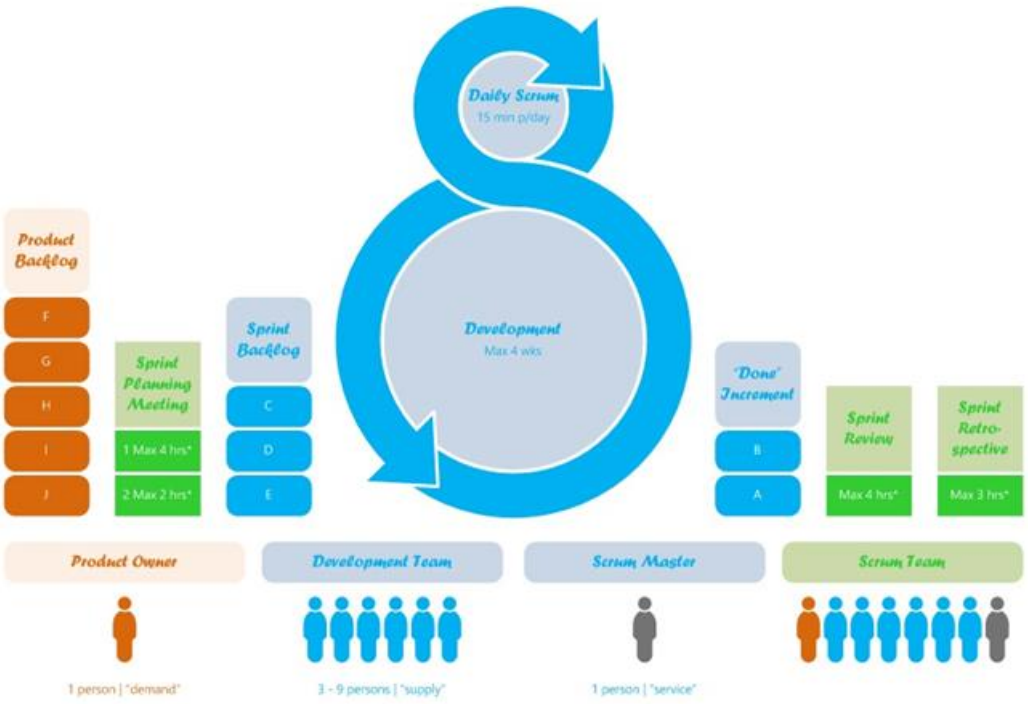
Reagování na změny před dodržováním plánu

V průběhu projektu může nastat potřeba změnit malé respektive velké části programu, tyto změny mohou být způsobeny změnou trendů na trhu nebo změnou zadání zákazníkem. Dodavatel by měl být schopný na nové požadavky reagovat v co nejkratším časovém úseku, aby se nevyvíjely zbytečné funkce, které nebudou mít pro zákazníka hodnotu a nebude je zřejmě ani chtít zaplatit. Smlouva o produktu by měla být rámcová a pružně reagovat na tyto případné změny. (10 str. 17)

1.5.2 Problémy vývoje a jejich řešení díky agilním metodám

Jak bylo popsáno výše, vývoj softwaru se potýká s mnohými problémy, které mohou vést až k neúspěchu projektů. Odpovědí na tyto problémy může být zavedení agilních metod. Konkrétní problémy a agilní metody, které je řeší, jsou zobrazeny v tabulce přílohy A.

1.6 SCRUM



Obrázek 4: Scrum koloběh

(Převzato z: https://calvinx.com/wp-content/uploads/2014/05/scrum-overview-mark-hoogveld-1024x724.jpg)

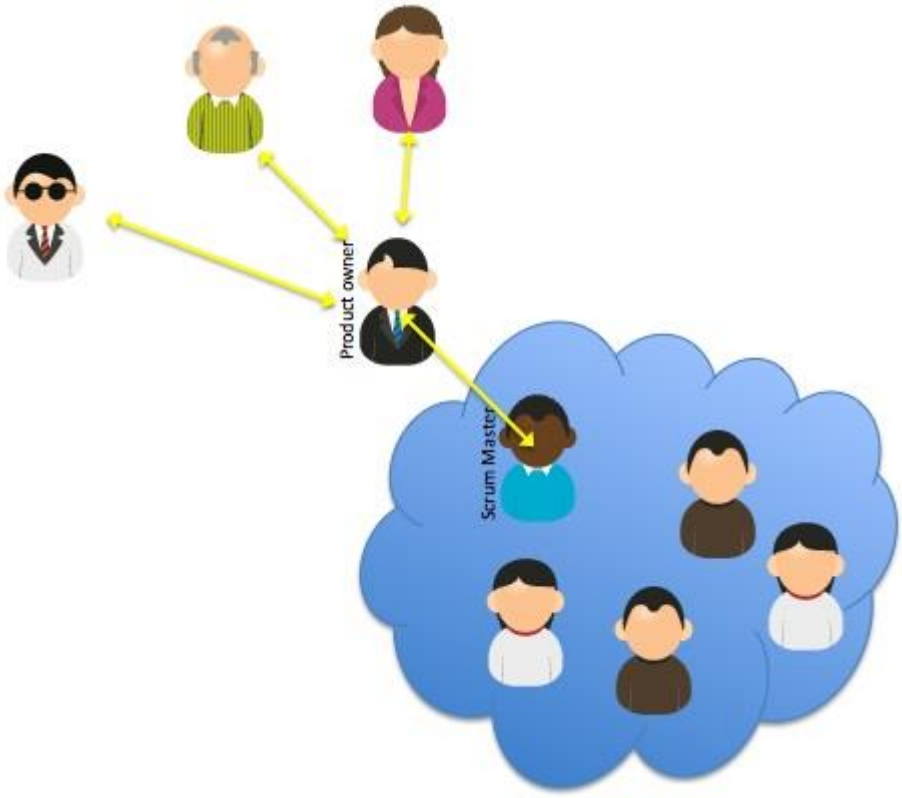
Vývoj podle metody Scrum probíhá v cyklech nazývaných Sprint. Jejich délka je nižší než jeden měsíc a je obvykle udávána v celých týdnech. Jednotlivé sprinty na sebe navazují a jejich délka je fixní, nikdy se neprodlužují, ať už práce v nich obsažená byla nebo nebyla dokončena. Práci na konci každého sprintu je považovat za dokončenou, když na ní proběhli testy, je plně integrovaná do systému a může být vydána. (9 str. 6)

Na začátku každého sprintu se sejde tým obsahující různé specialisty nutné pro vývoj, kteří určí podle priorit požadavky zákazníka, které se zavážou na konci sprintu dodat. Tento výběr se po odsouhlasení nesmí měnit. V průběhu sprintu se tým denně schází, aby zhodnotil dosavadní průběh a optimalizoval úspěšné

dosažení zvolených cílů. Po dokončení sprintu se tým sejde se stakeholdery, předvede, co bylo vytvořeno, získá zpětnou vazbu a zhodnotí průběh sprintu. (9 str. 6)

V dnešní době je používán malými tak i velkými firmami jako Microsoft, Google a dalšími. (9 str. 7)

1.6.1 Role



Obrázek 5: Scrum role a stakeholdři

(Převzato z: http://www.scrum-institute.org/images_scrum/Scrum_Roles_Stakeholders.jpg)

Správně fungující Scrum vyžaduje zavedení nových rolí, které v klasickém pojetí projektového řízení neexistují, navíc se mění pohled na některé klasické funkce a jejich odpovědnost. Pokud budou tyto odpovědnosti dodrženy, tým bude

komunikovat a držet spolu, potom splňuje všechny předpoklady pro úspěšné dokončení projektu.

Scrum Master

Scrum Master nenahrazuje v projektu týmového nebo projektového manažera, ale zabezpečuje, aby všechny procesy Scrumu probíhaly podle stanovených pravidel. Druhou jeho úlohou je starat se o potřeby týmu a vytvářet spojovací článek s vnějšími vlivy, které by mohli tým rozptylovat a snižovat jeho výkonnost. V neposlední řadě by se měl věnovat rozvíjení schopností týmu tak i jednotlivých členů. (9 str. 16)

Při sestavování týmu, který nemá s prací podle pravidel Scrumu moc zkušeností, lze očekávat větší zapojení Scrum Mastera do organizace a řízení týmu, avšak postupem času a díky jeho podpoře a radám by se měl stát samostatně organizovaným a své problémy si řešit sám. (9 str. 16)

Služby Scrum Mastera pro Product Ownera

- Vytváření technik pro efektivní správu Product Backlogu,
- pomoc s udržování konzistentního a přehledného Product Backlogu
- pomoc s plánováním projektu,
- třídění Product Backlogu pro určení priorit a maximální efektivitu,
- dohled nad dodržování agilních zásad,
- spolupráce při Scrum schůzkách. (8 str. 6)

Služby Scrum Mastera pro vývojový tým

- vedení k samostatně organizovanému týmu a zastupitelnosti jednotlivých členů,
- odstraňování problémů, které se mohou objevit v průběhu projektu,
- spolupráce při Scrum schůzkách,
- ochrana týmu před vnějšími vlivy,

- identifikace oblastí v organizaci týmu, kde nejsou zásady Scrumu používány, nebo jim nebylo zcela porozuměno a následující náprava. (8 str. 6-7)

Služby Scrum Mastera pro organizaci

- vedení a školení organizace při zavádění Scrumu,
- plánování integrace Scrumu v organizaci,
- vysvětlení pravidel Scrumu a jeho dopadu na vývoj produktu zaměstnancům a stakeholdrům,
- zvyšování produktivity týmu,
- spolu ostatními Scrum Mastery zvyšovat efektivnost použití Scrumu v organizaci. (8 str. 7)

Role Scrum Mastera je pro vytvoření úspěšného týmu a následného produktu klíčová. Měla by to být osoba, která dovede vnímat své okolí, komunikovat s ním a schopná koučovat ostatní členy týmu a moderovat nejen Scrum schůzky. Scrum Master by měl být schopný řešit problémy, které mohou vzniknout, v nejlepším případě je předvídat a předcházet jim. Dalo by se říct, že pokud vykonává svoji práci perfektně tak o něm není vědět, pokud by byl ale z týmu přeřazen, tak by se s jistým zpožděním začaly objevovat problémy, které by neměl kdo řešit. Dobrý Scrum Master může pracovat na jakékoli funkci například vývojář, tester, projektový manažer nebo designer, jen je nutné si uvědomit problém týkající se předchozích manažerských funkcí, že nyní týmu nepřikazují a ani mu nejsou nadřazení. Všeobecně také neplatí, že například nejlepší programátor bude nejlepší Scrum Master, v extrému se může stát, že touto cestou přijde organizace o dva lidi, nezíská dobrého Scrum Mastera a navíc ztratí nejlepšího programátora. (10 str. 31-33, 9 str. 16)

V menších společnostech často nastává potřeba spojit tuto roli s nějakou jinou, z důvodů snížení nákladů. Toto se všeobecně nedoporučuje, protože nastává konflikt mezi odpovědnostmi, kde povinnosti Scrum Mastera jsou zastíněny

pomyslně důležitější druhou rolí, čímž trpí efektivita týmu. Pokud není jiná cesta než spojení dvou rolí, nikdy by to nemělo být s Product Ownerem nebo rolí jejichž cíle a priority jsou v konfliktu s rolí Scrum Mastera. (10 str. 32)

Product Owner

Product Owner odpovídá za maximalizaci návratnosti investice ve vztahu k financím, ale také k strávenému času vývojového týmu určením, jaké funkce se budou ve sprintu přidávat. Rozhoduje se podle definovaných funkcionalit a jejich priorit v Product Backlogu, ty neurčuje sám, ale s pomocí zákazníka nebo jeho zástupce, dále uživatelů, softwarového architekta a případně ostatních Product Ownerů. Jelikož je podoba Product Backlogu jeho odpovědnost, ostatní mu mohou pouze radit a konečné rozhodují je na něm. (9 str. 14, 10 str. 33)

Product Owner se snaží maximálně porozumět produktu a svoji vizi komunikuje s týmem, managementem i se zákazníkem. Strávením velkého množství času se zákazníkem pozná jeho pracovní prostředí a je schopen rozhodnout, kde je pravá hodnota pro zákazníka. Musí tedy najít vhodnou rovnováhu mezi poznáváním zákazníka a časem kdy se věnuje týmu, obvykle tráví prácí se zákazníkem 80 % času a zbytek věnuje týmu. Produt Owner však nikdy nikomu nepřikazuje, na čem má pracovat a kdy dokončit, pouze stanovuje co a v jakém pořadí se má dělat pomocí jím určených priorit. (10 str. 34)

Odpovědnosti Product Ownera týkající se Product Backlogu

- Úplný popis položek,
- Řazení Product Backlogu, pro nejlepší dosažení cílů,
- zajištění dostupnosti a jasnosti Product Backlogu a toho na čem bude tým pracovat v budoucnu,
- zajištění, že vývojový tým chápe obsah položek v dostatečné hloubce. (8 str. 5)

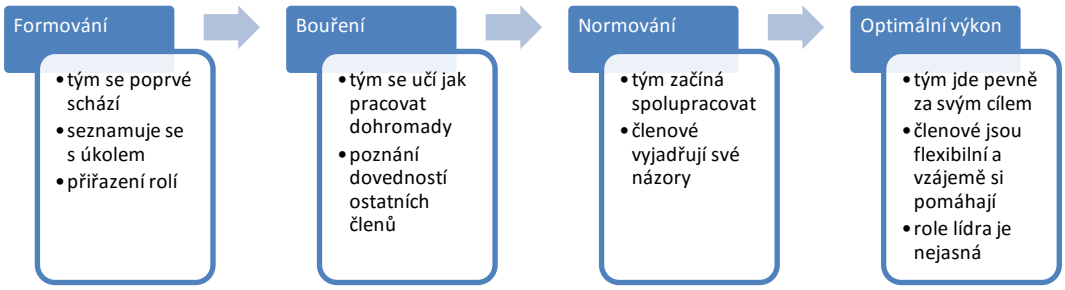
Product Owner Proxy

Ve velkých korporacích s distribuovanými týmy může nastat komplikace, kdy se Product Owner nachází od týmu příliš daleko a třeba i v jiném časovém pásmu. Řešním je volba jednoho člena svého týmu, který má k prostředí zákazníka a produktu velmi blízko, stává se z něj Product Owner Proxy. Tento člověk plně rozumí celému systému a je v denním kontaktu s týmem, přesto ale stojí na straně byznysu. Při práci s týmem musí být schopen nejen plně odpovědět na jejich dotazy, ale také činit rozhodnutí a nést za ně odpovědnost, je to plnohodnotný člen týmu Product Ownera. (10 str. 39)

Tým

Agilní vývojový tým spoléhá na jednotlivé členy a využívá adaptivní proces, který jednotlivý členové týmu mohou podle potřeby ovlivňovat a měnit. Podmínkou pro vznik takového self-organized týmu je společný cíl, tedy porozumění zákazníkovi a sjednocení vize určující jak má výsledek práce vypadat. Dalším důležitým prvkem je důvěra nejen mezi členy týmu, ale také k zákazníkovi a zbytku firmy. Týmu musí být umožněno, aby mohl měnit způsoby své práce a možnost podílet se na tvorbě Product Backlogu a přispět nápady, aby výsledek byl nejlepší možný. Pokud vznikne nějaký problém nebo se nepodaří stihnout všechny úkoly ve sprintu, je to chyba celého týmu. (10 str. 34-36)

Při založení nového týmu je nutno brát na zřetel, že tým nebude fungovat na plné obrátky hned od začátku, protože musí postupně projít fázemi popsanými v Tuckmanově modelu: formování, bouření, normování a optimální výkon. Doba vzniku plně funkčního týmu záleží na jednotlivcích, ale většinou trvá okolo tří sprintů, než je tým dostatečně vyspělý, aby dodával výsledky předvídatelně. (9 str. 15)



Obrázek 6: Tuckmanův model

(Zdroj: 10 str. 107-110)

Individuální členové týmu mají různé specializace a zaměření, k dosažení nejlepšího možného výsledku by bylo nejlepší, aby měl tým vyrovnaný soubor dovedností a tak byl schopen řešit neustále se měnící podmínky a byl co nejsamostatnější. V ideálním případě bude od začátku složen ze zástupců různých disciplín od začátku, ale každý člen se bude snažit učit od ostatních, aby mohl v případě potřeby zastoupit kteréhokoliv z ostatních členů týmu. (10 str. 36)

Tým by se měl skládat z programátorů, designérů uživatelského rozhraní a testerů v rozsahu pěti až devíti členů bez Product Ownera a Scrum Mastera, menší týmy nepřináší dostatečný nárůst produktivity, respektive u větších týmů by bylo potřeba příliš mnoho koordinace, aby tým mohl fungovat optimálně. (8 str. 6)

Zákazník

V agilním procesu vzniká snaha zapojit zákazníka do vývoje, aby určoval svoje priority, podílel se v průběhu projektu na jeho změnách a návrhu funkcionality, v podstatě se stal součástí týmu. Za zákazníka může být považována kdokoli se zájmem na projektu a to buď zevnitř firmy, nebo z jejího venkovního okolí. Jedná se o dlouhodobou spolupráci počínající seznámením se s pravidly Scrumu. Prezentací výsledků po dokončení každého sprintu zákazník vidí, že se něco děje a seznámí se s vývojáři. Pokud nastane technický problém, je spíše nakloněn

pochopení, než když je mu to oznámeno na konci termínu projektu. Na konec s přístupem do Product Backlogu a jednotlivým User Stories se stává opravdovým partnerem týmu. (10 str. 37)

Nejdůležitější je zákazníka transparentně informovat o úspěchu i vzniklých problémech, snažit se nedodat pouze produkt jaký si objednal, ale takový jaký doopravdy potřebuje. Pokud má takový partnerský vztah fungovat, musí pro něj existovat podmínky, mezi které patří ústupek zákazníka z klasických smluvních pravidel, jako pevně stanovené datum dodání nebo rozpočet. Také smlouva obsahuje základní funkcionality a ostatní se dodefinuje postupně pomocí priorit a ve chvíli kdy už nechce produkt dál vyvíjet, má právo od dalšího sprintu odstoupit. Pokud zákazník nechce ani v jednom ustoupit, nelze s ním spolupracovat na základě výše uvedených postupů. (10 str. 38)

Manažeři

Přestože se může zdát, že samostatně se organizující týmy nepotřebují manažera, opak je pravdou. Odpovědnost se přesouvá od řízení denních operativ, alokace zdrojů a řešení každodenních problémů k vytváření vhodného prostředí pro fungování týmu a podporu jednotlivých rolí. V menších firmách mívá na starosti koučování Scrum Masterů nebo produktovou strategii – tedy koučování Product Ownerů. Manažeři jsou posunuti výše ke strategickým rozhodnutím a ostatní delegují nižším rolím a týmům. (10 str. 39,40)

Projektoví manažeři i nadále obstarávají všechny náležitosti projektu, jako reportování průběhu do všech systémů společnosti, sledování a vyplňování rozpočtů a odpracovaného času na projektu. Dále taky sledování milníků a upozorňování týmů na ně. Do řízení týmu se však nezapojují a nepřidělují nikomu úkoly. (10 str. 41)

1.6.2 Artefakty

User Story

„Jako Uživatel chci Funkcionalitu, abych dostal Business hodnotu."

Toto je základní formát User Story, který říká nejen to, co se má dělat, ale také pro koho, a z jakého důvodu. Měla by být jasně popsatelná, malá, vytvářet obrázek, nezávislá, přinášet hodnotu. Mělo by být na první pohled zcela jasné, zda do ní má smysl investovat peníze a úsilí. Pokud User Story tyto podmínky nesplňuje, je těžké určit co je jejím obsahem a ani vývojový tým s ní nemůže pracovat a potřebuje časté konzultace. Taková User Story, které tým nemůže uchopit a nelze je podrobněji specifikovat se nazývají Epic User Story a lze je obvykle rozložit na několik podrobnějších User Story. Jsou tedy takovým souhrnným prvkem souvisejících User Story. (9 str. 45,46)

Každá User Story obsahuje také seznam podmínek, které musí být splněny, aby byla User Story přijatelná jako splněná. Tento seznam se nazývá akceptační kritéria a jsou definovány Product Ownerem spolu s vývojovým týmem a to nejpozději při plánování sprintu. (9 str. 46)

Sprint

Iterace je základním stavebním kamenem agilních metod, ve Scrumu je pojmenovaná Sprint. Díky pravidelnosti cyklů jsou výsledky prezentovány včas a vzhledem k plánování také přinášejí očekáváný výsledek. Samozřejmě toto platí pro zaběhnutý Scrum, v počátku zavedení metody nemusí k takovým výsledkům vždy dojít. Délka sprintu by měla být jeden až čtyři týdny podle rozsahu projektu, aby byl tým schopen během sprintu dokončit funkcionalitu a na konci ji prezentovat. Pokud se sprint nastaví zbytečně dlouhý, tým přijde o zpětnou vazbu jak od zákazníka, tak i interní od svých členů. Délku může také ovlivnit druh zákazníka nebo jeho povaha, pokud bude často měnit zadání, je vhodné zvolit kratší délku sprintů, která umožní rychle reagovat na jeho nové požadavky. (9 str. 10, 10 str. 43)

Product Backlog

V každém projektu je nutné někam zaznamenat, co všechno se bude vytvářet. Jeden z použitelných formátů k popsání funkcionality je User Story, použít je možné i jiný způsob, ale vždy by se mělo jednat o nezávislé funkční celky dodávající hodnotu zákazníkovi. Položky by mělo možné prezentovat a získat tak zpětnou vazbu. (7 str. 19)

Každá položka by měla obsahovat odhad náročnosti vytvořený týmem a prioritu. Podle priority by se také měla lišit podrobnost rozpracování konkrétních User Story. Ty s nejvyšší prioritou by měly být rozpracovány do nejpodrobnějších detailů, aby naplnily dva až tři sprinty. Položky se střední prioritou mají být rozpracované na menší kusy, které nemusí být detailně rozpracované, avšak Product Owner by měl mít představu o jejich funkcionalitě a měly by obsahovat práci na pět a více sprintů. Zbytek Backlogu není potřeba jakkoliv podrobně definovat a zůstává v podobě Epiků. (10 str. 46)

Sprint Backlog

Je malou součástí Product Backlogu a obsahuje funkcionalitu s vysokou prioritou, kterou se tým zavázal dodat ve Sprintu. Obsah si vybírá tým sám podle Product Ownerem určeným priorit, při plánování sprintu. Tým si často v průběhu sprintu rozloží jednotlivé User Story na konkrétní úkoly, čímž získá větší přehled o rizicích a statusu dodání celého závazku. Tyto úkoly jsou však pouze interní praktikou týmu, Sprint Backlog obsahuje výhradně User Story. (10 str. 47,48)

1.6.3 Meetingy

Scrum Meeting

Neboli Standup Meeting probíhá každý den ráno s maximální délkou deset minut. Důvodem schůzky je vzájemné sdělení mezi členy týmu co dělali včera, na čem budou pracovat dnes a na jaké narazili problémy. Schůzky se účastní pouze vývojový tým, Scrum Master má pouze roli moderátora. Meeting by měl probíhat

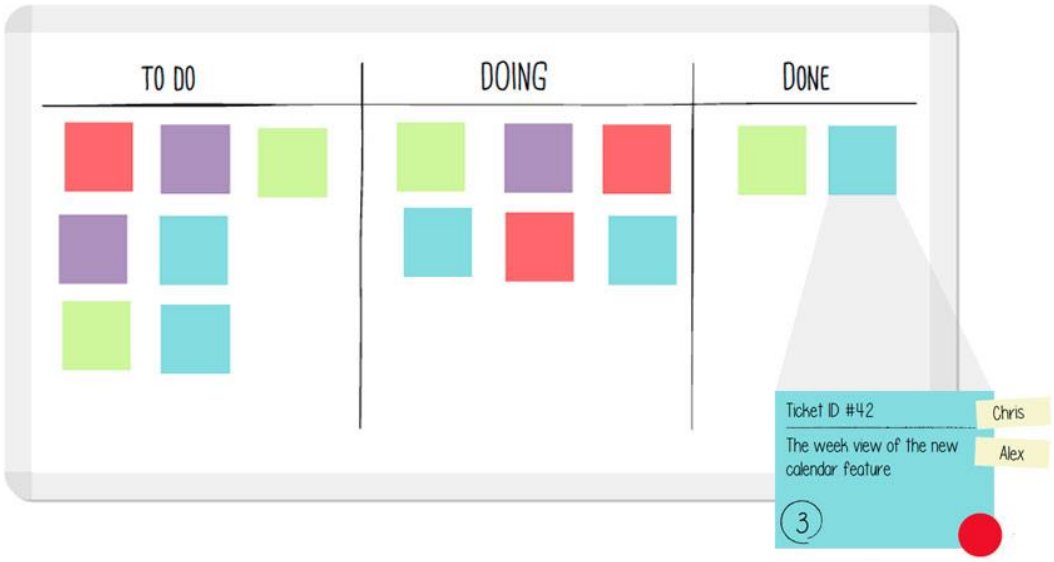
mimo stanice programátorů ve stoje u tabule s přehledem sprintu, tak aby se zabránilo zbytečnému protahování. (9 str. 10)

Sprint Review a Sprint Restrospective

Po doběhnutí sprintu přichází na řadu Sprint Review, této schůzky se účastní vývojový tým, Scrum Master, Product Owner, zákazníci, odborníci, vedení a kdokoliv z ostatních stakeholderů. Cílem schůzky je předvést zákazníkovi demo toho co se udělalo, získat vzájemnou zpětnou vazbu a rozhodnout co se bude dít dál, pokud je nutné provést nějaké změny. (9 str. 10)

Po Sprint Review následuje Sprint Retrospective, na kterém se sejde celý Scrum tým a prodiskutuje co průběh celého sprintu, co se v něm podařilo, co nefunguje. Pokud se tým shodne, že by šlo něco udělat jinak, jsou změny promítnuty v následujícím sprintu. (9 str. 10)

1.7 Kanban



Obrázek 7: Kanban tabule

(Převzato z: http://leankit.com/learn/wp-content/uploads/2015/11/kanban-board-e60650d1-1.jpg)

Kanban nepatří přímo mezi agilní metody je však často využíván ve Scrumu pro omezení rozpracovaných User Stories, jejich rychlé dokončení a vizualizaci stavu sprintu na tabuli. (16 str. 10)

Kanban pochází z Japonska, kde byl využíván pro řízení počtu lidí v chrámu pomocí lístečků. Vstoupit mohl pouze člověk s lístečkem, který při odchodu zase odevzdal, tak se nemohlo do chrámu dostat více lidí, než byla jeho kapacita. Princip následně využily japonské továrny pro dodávání dílů „just in time". (10 str. 105)

V IT je Kanban často používán pro udržování produktu v po projektové fázi vývoje. Vznikají problémy, protože Kanban nic nenařizuje a pravidla si měla určit společnost sama. Měly by se dodržovat tyto principy:

- Omezit rozpracovanou práci
- Minimalizovat čas průchodu
- Vizualizovat průběh (10 str. 105)

Kanban rozděluje proces na jednotlivé fáze, mezi kterými se přesouvají kartičky znázorňující úkol podle toho, v jaké části se nacházejí. Čistý kanban je zpravidla doplňován prvky agilních metod jako rolí Product Ownera, Standup Meetingem, Backlogem, User Story a dalšími. (16 str. 10)

2 Analýza současného stavu

2.1 Shrnutí

Společnost se zabývá softwarovým vývojem a v současné době provozuje tři středně velké, až velké informační systémy.

Hlavním produktem společnosti je informační systém, který vznik před deseti lety a byl postupně rozšiřován. Následkem vznikl velký procedurální kolos plný potenciálních rizik ukrytých v dlouhých funkcích. Systém používají desítky zákazníků, z toho plyne poměrně velké množství dovývojů, úprav a oprav vznikajících chyb.

Kvůli stavu tohoto informačního systému a náročnosti případné opravy se současným provozem bylo rozhodnuto vytvořit novou generaci tohoto informačního systému. Vývoj již probíhá a jsou vytvořeny základní ovládací prvky a nástroje, bohužel dosavadní průběh neodpovídá představám společnosti.

Další dva informační systémy byly vyvinuty v nedávné době, jejich technický stav odpovídá dnešním standardům psaní kódu a objektově orientovaného softwarového vývoje, využívají nástroje podporující ladění a kontrolu metrik. Dovývoj těchto systémů probíhá řádově v rozsahu deseti hodin měsíčně a větší úpravy jednou za čtvrt roku. Způsob fungování těchto projektů není nutné měnit.

3 Návrh řešení a přínos návrhů řešení

3.1 Kanban

V první fázi ve společnosti bude zaveden Kanban, který umožní připravit se na spuštění Scrumu a přitom zároveň pokračovat ve vývoji nové i staré generace informačního systému, každý projekt však bude mít svůj vlastní. Kanban bude využívat prvky Scrumu jako User Story, denní Scrum Meeting, User Story a Product Backlog. Jelikož je velmi podobný současnému způsobu zpracovávání úkolů, vývojový tým dostane možnost postupně se seznámit s novými procesy.

Jednotlivé fáze procesu budou:

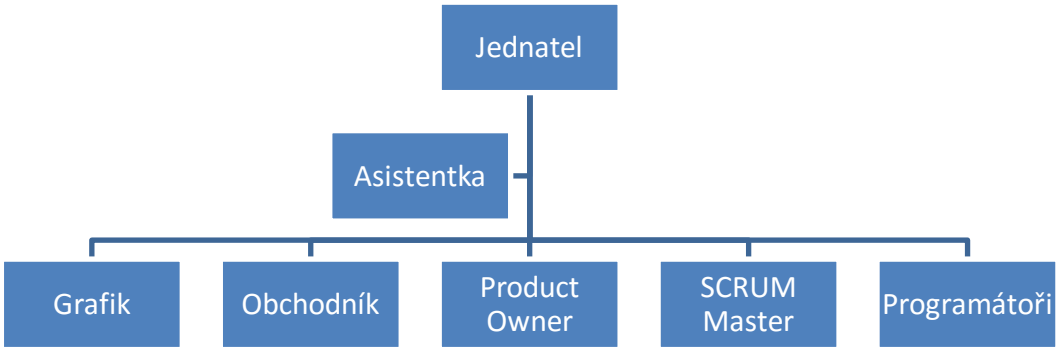
- Backlog – zásobník úkolů, které byly vybrány k vývoji
- Vývoj – právě probíhají práce na těchto úkolech
- Testování – úkoly v této kolonce jsou hotovy a čekají na otestování, jehož součástí je i kontrola kvality kódu
- Hotovo – hotové a otestované úkoly
- Verze – v této kolonce budou hotové otestované úkoly, které budou zahrnuty v příští verzi informačního systému. Zahrnutím do verze úkoly vystupují z procesu.

K zobrazení průběhu nebude využíván žádný speciální software, ale pouze tabule s barevnými lístečky. Lístečky budou barevně rozlišeny podle typu úkolu na chyby, nový vývoj a úpravy. Priorita bude mít pouze jednu úroveň a úkoly budou speciálně označeny. Ve chvíli kdy si programátor vybere úkol, označí ho svou značkou.

Po zavedení Scrumu pro vývoj nové generace informačního systému zůstane Kanban využíván pro vývoj staré verze.

3.2 Transformace uspořádání společnosti

Aby mohla společnost využívat postupů Scrumu je nutné zavést některé nové role. Scrum Masterem se stane jeden z programátorů, který se nyní stará o jeden z dalších informačních systémů. Kvůli velikosti kolektivu však bude i nadále působit jako programátor, starat se o tento informační systém a primárně pomáhat s úkoly na staré generaci informačního systému. Povinnostem Scrum Mastera by se měl věnovat primárně a to minimálně 4 hodiny denně, podle potřeby i víc.



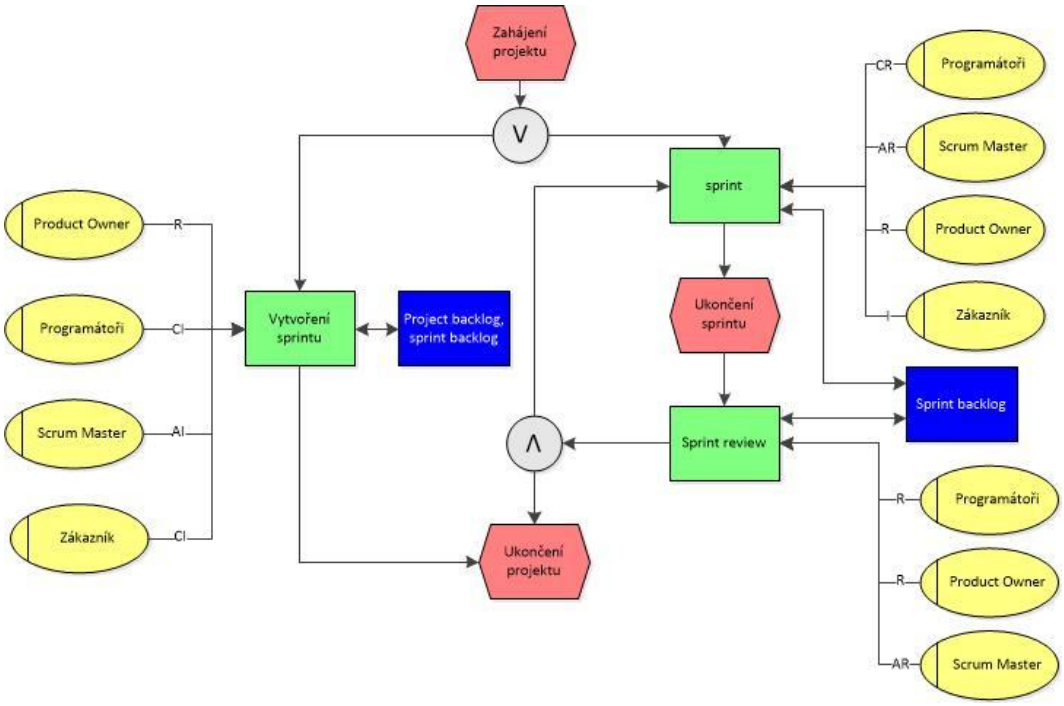
Graf 1: Nová struktura společnosti

(Zdroj: Vlastní)

Povinnosti Product Ownera bude provádět dosavadní Project Manager, většina jeho dosavadních povinností přechází na self-organized vývojový tým. Z důvodu velkého množství zákazníků není možná obvyklá spolupráce s jedním zákazníkem, místo toho využívá společnost zpětnou vazbu od uživatelů ze školení. Plánování priorit a vytváření User Story probíhá se školiteli, obchodníkem a zástupci dvou největších zákazníků. Toto řešení umožní reagovat na zpětnou vazbu od uživatelů denně pracujících s informačním systémem, nabídku funkcí konkurence a požadavky klientů.

3.3 Scrum

Hlavním podnikovým procesem se stává jeden iterační interval vývoje projektu, sprint. Obsah sprintu je definován Product ownerem a Scrum Masterem podle priority pro zákazníka, návaznosti na předchozí a následující sprinty a v neposlední řadě tak, aby výsledkem sprintu byl celek funkčností, které je možné předvést zákazníkovi. Tím tým získává zpětnou vazbu a případná oprava chyb může být začleněna do následujícího sprintu nebo jiného budoucího. Po ukončení sprintu následuje zhodnocení průběhu, a podle toho možné korekce dalších sprintů, jejich obsahu nebo délky, pokud je to nutné.



Graf 2: EPC diagram Scrum procesu

(Zdroj: Vlastní)

EPC diagram popisuje zhotovení projektu agilní metodou Scrum. Hlavní proces je série intervalů vývoje – sprintů. Podpůrnou činností je vytváření a plánování těchto sprintů a po jejich dokončení zhodnocení průběhu a poznatků. Ke každému

procesu jsou přiřazeni hlavní aktéři, podrobněji jsou aktéři popsáni v RACI matici.

3.3.1 RACI matice

R - Responsible - kdo je odpovědný za vykonání svěřeného úkolu

A - Accountable - kdo je odpovědný za celý úkol, je odpovědný za to, co je vykonáno

C - Consulted - kdo může poskytnout cenou radu či konzultaci k úkolu

I - Informed - kdo má být informován o průběhu úkolu či rozhodnutích v úkolu

Tabulka 8: RACI matice

(Zdroj: Vlastní)

RACI matice	Role					
	Jednatel	Grafik	Product Owner	Scrum Master	Programátoři	Zákazník
Analýza požadavků zákazníka			AR		C	I
Scrum sprint	I	C	R	AR	CR	I
Oprava chyby		C	A		R	CI
Návrh grafického designu		AR			C	CI
Vytvoření sprintu			R	AI	CI	CI
Scrum meeting			I	AR	R	
Sprint review	I		R	AR	R	
Backlog grooming		C	AR	R	R	

3.3.2 Délka sprintu

V případě, že se jeden z programátorů bude věnovat roli Scrum Mastera polovinu své pracovní doby sníží se počet vývojářů na 6,5 programátora. Sprint bude na počátku nastaven na délku dvou týdnů, s celkovým fondem 520 hodin při osmihodinové pracovní době. Úkoly budou plánovány na pět hodin denně, vytvoří se tak poměr dvě třetiny času na nový vývoj a zbytek na ostatní činnosti.

Tabulka 9: Hodiny ve dvoutýdenním Scrumu

(Zdroj: Vlastní)

Hodin denně	2 týdenní Scrum
5	325
5,5	357,5
6	390

Pokud budou User Story vyžadovat větší množství práce, může se délka sprintu protáhnout na dobu tří týdnů a zvýšit tak hodinový fond na 780 hodin. Další možností je zvýšit délku denního Scrum úkolu na úkor ostatních úkolů.

Tabulka 10: Hodiny v třítýdenním Scrumu

(Zdroj: Vlastní)

Hodin denně	3 týdenní Scrum
5	487,5
5,5	536,25
6	585

3.4 Dělení User Story na úkoly

Stěžejním problémem je naučit se dělit User Story na malé úkoly, ale také určit všechny úkony nutné ke splnění a odhadnutí jejich časové náročnosti. Dělit User Story na pětihodinové úkoly nebude v některých případech možné a nemělo by se jakkoliv násilně dělit „aby to vyšlo". Pokud bude úkol delší než stanovená délka denního Scrum úkolu, programátor bude pracovat na zadání, dokud nebude splněno a poté na úkolech na starém systému, aby dodržel poměr práce mezi vývojem na novém a starém informačním systému.

3.5 Přechod mezi Kanbanem a Scrumem tam i zpět

Přechod z Kanbanu na SCRUM by měl proběhnout, jakmile jsou připraveny User Story a s nimi související úkoly na dva sprinty, všechny probíhající úkoly Kanbanu jsou řádně ukončené bez chyb, je vydána nová verze obsahující tyto poslední úpravy a vývojový tým je dostatečně stabilní, aby mohly sprinty v pořádku probíhat.

V případě, pokud by týmu nevyhovovalo využívat praktiky SCRUMU a chtěl by se vrátit ke Kanbanu jedinou podmínkou je dokončený sprint. Jedním z dalších možných důvodů, který by mohl vést k návratu ke Kanbanu je nestabilní složení vývojového týmu, které znemožňuje plánování sprintů.

3.6 Analýza rizik

Pro hladké provedení změn je potřeba analyzovat rizika ohrožující průběh této změny pomocí metody RIPRAN. Nejdříve je nutné definovat hodnocení dopadu a pravděpodobnosti rizik.

Tabulka 11: Hodnocení pravděpodobnosti a dopadu

(Zdroj: Vlastní)

	Dopad	Pravděpodobnost
(0;1>	Bezvýznamný	Nepravděpodobné
(1;2>	Významný	Pravděpodobné
(2;3>	Drtivý	Vysoce pravděpodobné

Brainstormingem a dalšími metodami byla identifikována možná rizika a jejich scénáře, které byly ohodnoceny z hlediska dopadu a pravděpodobnosti, čímž byla zjištěna hodnota těchto rizik.

Tabulka 12: Analýza a kvantifikace rizik

(Zdroj: Vlastní)

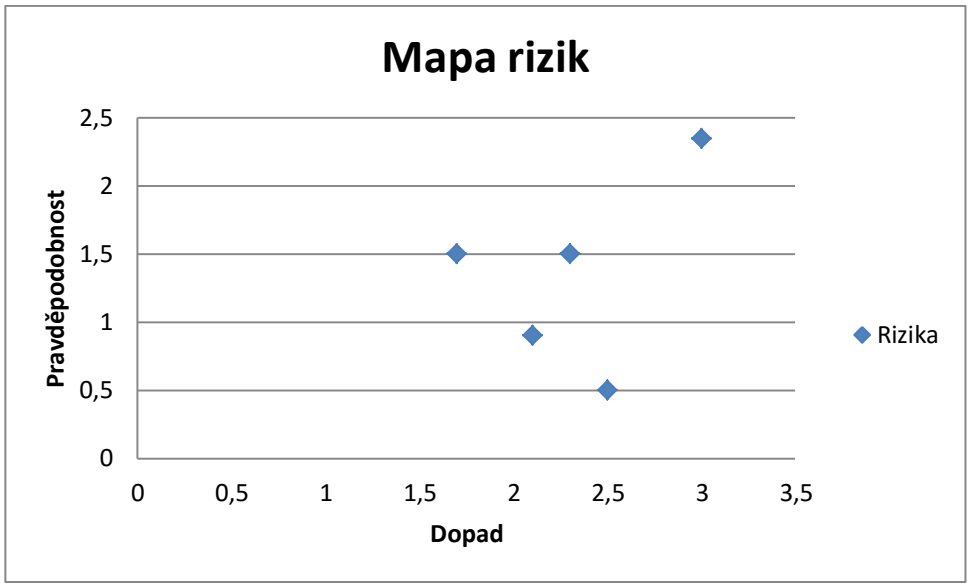
Pořadí	Hrozba	Scénář	Pravděpodobnost výskytu	Dopad	Hodnota rizika
1	Špatné ohodnocení User Story	Prodloužení sprintu	2,35	3	7,05
2	Nedostatečná komunikace se zákazníkem	Snížení kvality projektu	1,5	1,7	2,55
3	Kolize jednotlivce od týmové spolupráce	Rozpad týmu	0,5	2,5	1,25
4	Špatný průběh sprintu	Zpoždění projektu	1,5	2,3	3,45
5	Nedostatečná komunikace během sprintu	Zpoždění projektu	0,9	2,1	1,89

Po ohodnocení rizik bylo vytvořeno slovní ohodnocení rizik a rizika byla znázorněna v mapě rizik.

Tabulka 13: Hodnota rizik

(Zdroj: Vlastní)

Hodnota rizika	
(0;2>	Běžné
(2;5>	Závažné
(5;9>	Kritické



Graf 3: Mapa rizik

(Zdroj: Vlastní)

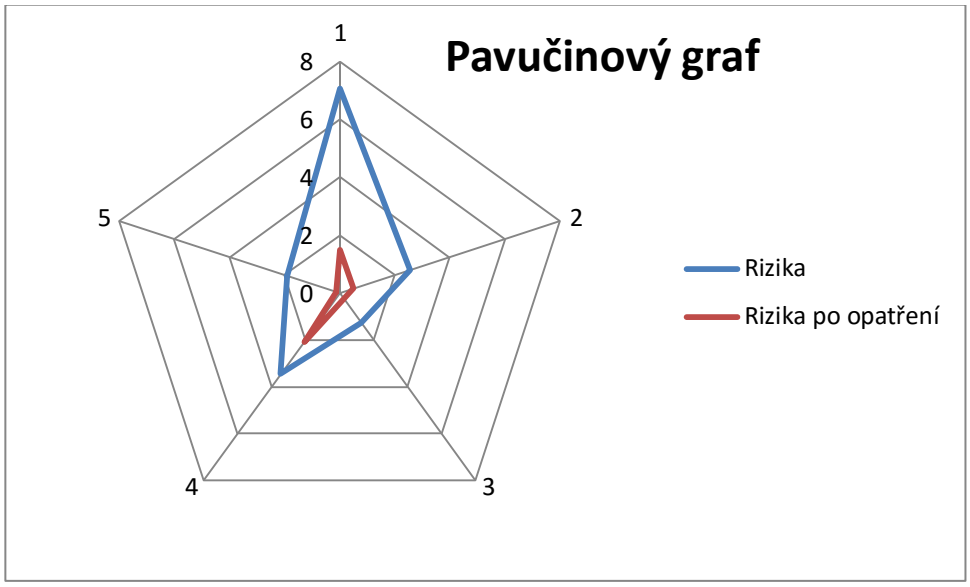
Po nalezení a vyhodnocení rizik je nutné vytvořit opatření zmírňující dopad respektive pravděpodobnost rizik, znovu vypočítat hodnotu rizika určit náklady a zodpovědnou osobu.

Tabulka 14: Opatření pro snížení rizik

(Zdroj: Vlastní)

Pořadí	Opatření	Pravděpodobnost výskytu	Dopad	Hodnota rizika	Náklady	Zodpovědnost
1	Hodnocení všemi členy týmu	0,5	3	1,5	4 hod	Scrum Master
2	Pravidelné schůzky product ownera se zákazníkem	0,5	1	0,5	2 hod	Product Owner
3	Dodržování pravidel agilního vývoje	0,1	2,5	0,25	-	Scrum Master
4	Sprint review meeting	0,9	2,3	2,07	4 hod	Scrum Master
5	Denní Scrum meetingy	0,1	1,1	0,11	10 min	Scrum Master

Navržená opatření snižují hodnotu rizik, velikost snížení přibližuje pavučinový graf. Obecně lze říci, že pokud bude společnost a její zaměstnanci dodržovat pravidla agilního přístupu, k těmto rizikům by nemělo nikdy dojít.



Graf 4: Pavučinový graf

(Zdroj: Vlastní)

3.7 Náklady na zavedení

Přestože je možné provozovat funkční Kanban a Scrum s pouze minimální počáteční investicí, je třeba zvážit investici do certifikačních kurzů, které předají osobám, kterým byly přiděleny stěžejní role hlubší informace. Pokud nebude Scrum fungovat a nepomohou ani školení hlavních rolí, je možné pozvat do firmy experta na agilní vývoj, a s jeho pomocí napravit problémy s procesem. Mezi potřeby pro zobrazování průběhu činností patří bílá popisovací magnetická tabule o rozměrech 1,8×1,2 metrů dále barevné nalepovací lístečky, magnety, lihové fixy a čistící prostředky na tabuli.

Mezi náklady je nutné také započítat dočasné snížení efektivnosti, než si vývojový tým zvykne na nové postupy a procesy.

Tabulka 15: Náklady na zavedení

(Zdroj: Vlastní)

Položka	Cena
Certifikace Scrum Master	30 400 Kč
Certifikace Product Owner	30 400 Kč
Potřeby pro zobrazování průběhu činností	4 500 Kč
Celkem	**64 300 Kč**

3.8 Přínos řešení

- **Přehlednost** – metody přinesou pravidla vytváření, zadávání a odevzdávání úkolů. Získá se také přehled o tom kdo, na čem aktuálně pracuje, kdy bude úkol přibližně hotov, jakou práci bude dělat poté a co bude náplní blízké budoucnosti.
- **Zkvalitnění analýzy** – dodržováním určených postupů nedojde k vzniku úkolů, na kterých neproběhla analýza. V případě nedostatečných informací nepřibere Scrum tým User Story do sprintu během plánování a upozornění Product Ownera o nekompletnosti. Tým si během plánování sprintu sám určí, jaké činnosti souvisí s konkrétní funkčností, a protože se provádí v týmu, je větší šance, že se na nic nezapomene.
- **Zefektivnění vývoje** – zintenzivnění spolupráce mezi vývojáři a obchodníky přinese jasnější určení směru vývoje, tedy bude se vyvíjet to, co chce zákazník a ne funkcionalita, kterou si programátoři myslí, že chce nebo ji dokonce chtějí oni. Z kvalitnější analýzy také plyne rychlejší vývoj bez překvapení, na která nikdo nepomyslel.
- **Zvýšení morálky týmu** – s pomocí nástrojů zobrazujících průběh vývoje jako Burndown Chart a Kanban tabule uvidí programátoři průběh své práce a růst produktu. Zavedení nových praktik by také mělo tým

vytrhnout z případné letargie. Vývojáři získají možnost účastnit se plánování vývoje a projevit názory.

- **Zvýšení zastupitelnosti** – vývojáři si vybírají úkoly sami, není zde tlak projekt manažera, aby byl úkol splněn co nejrychleji, a tak předán programátorovi s největšími zkušenostmi s daným problémem.

4 Závěr

Práce je rozdělena na tří hlavní části. První obsahuje teoretická východiska práce seznamující s projektovým managementem a specifickými vlastnostmi projektů informačních technologií. Protože velké procento projektů nesplní zadání nebo jsou ukončeny před dokončením, práce zkoumá faktory, které jsou možným důvodem neúspěchu. Použitá studie také určuje faktory, které snižují riziko neúspěchu. Dále byly popsány klasické modely projektového řízení, mezi které patří Vodopádový, Spirálový a Iterativní model. Následuje podrobný popis Scrumu, jeho jednotlivých rolí, artefaktů a meetingů. Závěrem této části je zmíněn Kanban, který nepatří mezi agilní metody, přesto s nimi úzce souvisí.

Druhá část se zabývá společností, ve které by mělo dojít k zavedení agilních metod. Analýza se zaměřuje na provozované produkty a jejich technologický stav, dále rozebírá vývojové procesy společnosti. Z analýzy vyplívá, že jeden tým vývojářů udržuje stále hojně používaný informační systém a souběžně vyvíjí jeho novou generaci.

Řešení navrhuje použití Kanbanu pro řízení prací na staré verzi, ale také v počátku vývoje nové generace než dojde k naplnění podmínek pro přechod na Scrum. Návrh se dál zabývá zvolením vhodného časového rozsahu sprintu a stanovením vhodného množství práce během sprintu. Po provedení analýzy rizik bylo zjištěno, že nalezeným rizikům bude předejito dodržováním zásad agilního vývoje a Scrumu. Závěrem návrhu jsou zmíněny náklady na zavedení a přínosy řešení.

Cílem práce byl návrh řešení řízení projektu vývoje nové generace informačního systému a projektu v poprojektové fázi. Navržené řešení je aktuálně zaváděno ve společnosti, čímž bylo splnění cíle potvrzeno.

Seznam použitých zdrojů

1. BEECHAM, Sarah, John NOLL a Ita RICHARDSON. Using Agile Practices to Solve Global Software Development Problems - A Case Study. *2014 IEEE International Conference on Global Software Engineeering Workshops* [online]. IEEE, 2014, 5-10 [cit. 2016-01-5]. DOI: 10.1109/ICGSEW.2014.7. ISBN 978-1-4799-5206-9. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6912078
2. BRUCKNER, Tomáš, et al. *Tvorba informačních systémů: principy, metodiky, architektury*. 1. vyd. Praha: Grada, 2012, 357 s. ISBN 978-80-247-4153-6.
3. DOLEŽAL, Jan, Pavel MÁCHAL a Branislav LACKO. *Projektový management podle IPMA*. 1. vyd. Praha: Grada, 2009, 507 s. Expert (Grada). ISBN 978-80-247-2848-3.
4. DOUCEK, Petr. *Řízení projektů informačních systémů.* Praha: Profesional publishing, 2004. 163 s. ISBN 80-86419-71-1.
5. IKONEN, Marko, Elena PIRINEN, Fabian FAGERHOLM, Petri KETTUNEN a Pekka ABRAHAMSSON. On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation. *2011 16th IEEE International Conference on Engineering of Complex Computer Systems* [online]. IEEE, 2011, 305-314 [cit. 2016-01-21]. DOI: 10.1109/ICECCS.2011.37. ISBN 978-1-61284-853-2. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5773404
6. PURNA SUDHAKAR, Goparaju. A model of critical success factors for software projects. *Journal of Enterprise Information Management* [online]. 2012, **25**(6), 537-558 [cit. 2016-01-5]. DOI: 10.1108/17410391211272829. ISSN 1741-0398. Dostupné z: http://www.emeraldinsight.com/doi/abs/10.1108/17410391211272829
7. RUBIN, K. S., Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley Professional,2012. 452 s. ISBN 978-01-370-4329-3.

8. SCHWABER, Ken a Jeff SUTHERLAND. *The Definitive Guide to Scrum: The Rules of the Game* [online]. 2013 [cit. 2016-01-10]. Dostupné z: http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf
9. SUTHERLAND, Jeff. *Scrum Handbook* [online]. 2010 [cit. 2016-01-21]. Dostupné z: http://jeffsutherland.com/scrumhandbook.pdf
10. ŠOCHOVÁ, Zuzana a Eduard KUNCE. *Agilní metody řízení projektů*. 1. vyd. Brno: Computer Press, 2014, 175 s. ISBN 978-80-251-4194-6.
11. WHITTAKER, Brenda. What went wrong? Unsuccessful information technology projects. *Information Management* [online]. 1999, **7**(1), 23-30 [cit. 2016-01-10]. DOI: 10.1108/09685229910255160. ISSN 0968-5227. Dostupné z: http://www.emeraldinsight.com/doi/abs/10.1108/09685229910255160
12. *Big Bang Boom* [online]. The Standish Group International, Inc., 2014 [cit. 2016-01-10]. Dostupné z: https://www.standishgroup.com/sample_research_files/BigBangBoom.pdf
13. *CHAOS MANIFESTO 2013* [online]. The Standish Group International, Inc., 2013 [cit. 2016-01-10]. Dostupné z: https://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf
14. *The Standish Group Report* [online]. Project Smart, 2014 [cit. 2016-01-10]. Dostupné z: https://www.projectsmart.co.uk/white-papers/chaos-report.pdf
15. THE AGILE ALLIANCE, 2001. *Manifest Agilního vývoje software* [online]. 2001 [cit. 2016-01-10]. Dostupné z: http://agilemanifesto.org/iso/cs/
16. AHMAD, Muhammad Ovais, Jouni MARKKULA, Markku OIVO, Petri KETTUNEN a Pekka ABRAHAMSSON. Kanban in software development: A systematic literature review. *2013 39th Euromicro Conference on Software Engineering and Advanced Applications* [online]. IEEE, 2013, , 9-16 [cit. 2016-01-21]. DOI: 10.1109/SEAA.2013.28. ISBN 978-0-7695-5091-6. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6619482

Seznam obrázků

Seznam tabulek

Seznam grafů

Seznam příloh

Příloha A – Problémy vývoje a jejich řešení díky Agilním metodám

Problém	Agilní metoda nebo nástroj
Řeší plně	
neshoda rolí a dovedností	self-organized tým
nejasné odpovědnosti	self-organized týmu, definování rolí
nutnost samostatnosti	self-organized tým
nemožnost projevit se, demotivace	self-organized týmu, Jednotlivci a interakce před procesy a nástroji
sdílení znalostí	scrum meeting
přesčasy	udržitelné tempo
nespecifikované procesy	SCRUM, XP a další metody
nekompletní/chybící požadavky	Product Owner, ohodnocení náročnosti, časté iterace
nereálné očekávání zákazníka	reagování na změny, spolupráce zákazníka
Řeší částečně	
nejasný/špatný psaný projev	Fungující software před vyčerpávající dokumentací
specifikace z nabídky	Fungující software před vyčerpávající dokumentací
napětí mezi vedením a vývojáři	self-organized tým
Neřeší	
nedostatečný výcvik	
omezený kariérní postup	
nadměrné cestování	

(Zdroj: 1 str. 10)